DirectDSP CTBus, DS3, and H.110 API Description

Signalogic, Inc

Rev 1.5

Jul 2009

1. Overview

This document describes TDM-related APIs contained in DirectDSP software. TDM data is supported in the following categories:

- CTBus interface to the SigC641x DSP Resource PTMC module
- Channelized data provided by the wan256T3 (DS3) PTMC module
- H.110 data provided by the T8110 device on the HW400c/2 cPCI carrier board

The term 'CTBus' refers to the TDM bus specified in the PTMC module standard (PICMG v2.15).

APIs described below are in two (2) categories: high-level and low-level. High-level APIs are used to set a general "mode" of operation, for example to place an H.110 interface into clock master or clock slave mode.

Low-level APIs are used to access individual devices on the PTMC modules or cPCI carrier board, for example write to a specific register in the TECT3 PM4328 framer device on the DS3 module.

2. InitH110(const char* szBoard, int nBoard, UINT uMode, KN_ERROR* const pError)

Description

Initialize H.110 interface on HW400c/2 board, including T8110 device and CPLD.

Example Usage

```
uH110Mode = 0;

if (ConfigInfo.fH110ClockMaster)
  uH110Mode = H110_CLOCK_MASTER_A;
else
  uH110Mode = H110_CLOCK_SLAVE;   /* default is to treat clock, framesyncs as input */

if (uH110Mode & H110_CLOCK_MASTER_A) {

   uH110Mode |= H110_CLOCK_SOURCE_NETWORK;      /* master mode:  use local side WAN */
   uH110Mode |= H110_CLOCK_FALLBACK1_LOCALOSC;  /* fallback to local osc if network fails */
}
else if (uH110Mode & H110_CLOCK_MASTER_B) {

   uH110Mode |= H110_CLOCK_SOURCE_NETWORK;      /* master mode:  use local side WAN */
   uH110Mode |= H110_CLOCK_FALLBACK1_LOCALOSC;  /* fallback to onboard osc if network fails */
}
else {

   uH110Mode |= H110_CLOCK_SOURCE_CTC8A;        /* slave mode */
   uH110Mode |= H110_CLOCK_FALLBACK1_CTC8B;     /* fallback to CTC8B line if CTC8A fails */
}

uH110Mode |= H110_CLOCK_RATE_8192;              /* H.110 clock rate 8.192 MHz */

hBoardH110 = InitH110("OSSH110", nBoard, uH110Mode, &knError);

if (!hBoardH110) {

   printf("Error initializing HW400c/2 H.110 circuitry, Board Num = %d, error status = &d\n",
          nBoard, knError);
}
```

Existing Code

```
appTest.cpp (look for “initialize H.110 devices” comment)
```

Parameters

szBoard Name of manufacturer and board.

nBoard Board number (zero based).

uMode Mode and attributes to initialize the H.110 interface on the HW400c/2 board. The following clock modes defined in tdmlib.h are allowed; these modes are mutually exclusive (only one can be used at one time):

```
H110_CLOCK_MASTER_A
H110_CLOCK_MASTER_B
H110_CLOCK_SLAVE
H110_CLOCK_STANDALONE
```

The following clock mode attributes defined in tdmlib.h are allowed; one or more of these attributes may be combined with mode values shown above:

```
H110_CLOCK_SOURCE_CTC8A       /* H110 CTC8A line (clock A 8.192 MHz) */
H110_CLOCK_SOURCE_CTC8B       /* H110 CTC8B line (clock B 8.192 MHz) */
H110_CLOCK_SOURCE_NETWORK     /* WAN, i.e. clock recovered from T3 line */
H110_CLOCK_SOURCE_NETREF1
H110_CLOCK_SOURCE_NETREF2

H110_CLOCK_FALLBACK1_NONE
H110_CLOCK_FALLBACK2_NONE
H110_CLOCK_FALLBACK1_NETWORK
H110_CLOCK_FALLBACK1_NETREF1
H110_CLOCK_FALLBACK1_NETREF2
H110_CLOCK_FALLBACK1_CTC8A
H110_CLOCK_FALLBACK1_CTC8B
H110_CLOCK_FALLBACK1_LOCALOSC
H110_CLOCK_FALLBACK2_NETWORK
H110_CLOCK_FALLBACK2_NETREF1
H110_CLOCK_FALLBACK2_NETREF2
H110_CLOCK_FALLBACK2_CTC8A
H110_CLOCK_FALLBACK2_CTC8B
H110_CLOCK_FALLBACK2_LOCALOSC

H110_CLOCK_RATE_8192
H110_CLOCK_RATE_2048
H110_CLOCK_RATE_1544
```

pError Error code for the function (valid only if return value indicates the function failed).

Return Value

Success or failure of the function (0 = Fail, 1 = Success).

Notes

Contains an implicit call to DirectDSP API DSAssignBoard().

3. InitDs3Module(const char* szBoard, int nModule, UINT uMode, KN_ERROR* const pError)

Description

Initialize wan256T3 DS3 module.

Example Usage

```
/* DS3 module config mode:  config LIU, some DS3 level settings of TECT3 framer, and T8110.
   In the Siglog application, T1 and E1 level settings in the TECT3 framer should be initialized
   by either a) NComm software, or b) external script */

   uDs3Mode = DS3_CONFIG_LIU | DS3_CONFIG_TECT3 | DS3_CONFIG_T8110;

/* LIU settings */

   uDs3Mode |= DS3_LIU_NO_LOOPBACK;

/* T8110 settings */

   if (fConfigFileFound) {

      if (ConfigInfo.nDs3ClockOutput == KN_CLKOUT_CT8) uDs3Mode |= DS3_T8110_CLKOUT_CT8;
      else uDs3Mode |= DS3_T8110_CLKOUT_NETREF;

   /* TECT3 framer settings */

      if (ConfigInfo.nDs3Framing != KN_FRAMING_IGNORE) {

         if (ConfigInfo.nDs3Framing == KN_FRAMING_CBIT)
           uDs3Mode |= DS3_TECT3_CBIT_FRAMING;
         else if (ConfigInfo.nDs3Framing == KN_FRAMING_M13)
           uDs3Mode |= DS3_TECT3_M13_FRAMING;
      }
   }

   hBoardDs3 = InitDs3Module("OSSDS3", nDs3Module, uDs3Mode, &knError);
```

Existing Code

```
appTest.cpp (look for “initialize DS3 module” comment)
```

Parameters

szBoard Name of manufacturer and board.

nModule Module number (zero-based).

uMode Attributes to initialize and/or configure devices on the wan256T3 DS3 module. The following attributes defined in tdmlib.h are allowed; one or more of these attributes may be combined as needed:

```
DS3_CONFIG_LIU
DS3_CONFIG_T8110
DS3_CONFIG_TECT3
```

Note: the following attributes “take effect” only if their corresponding device configuration attribute listed above is active. For example, DS3_TECT3_xx attributes shown below are ignored unless the uMode includes the DS3_CONFIG_TECT3 attribute shown above.

```
DS3_T8110_CLKOUT_CT8
DS3_T8110_CLKOUT_NETREF

DS3_LIU_NO_LOOPBACK
DS3_LIU_ANALOG_LOCAL_LOOPBACK
DS3_LIU_DIGITAL_LOCAL_LOOPBACK
DS3_LIU_DIGITAL_REMOTE_LOOPBACK
DS3_LIU_TRANSMIT_ALL_1S
DS3_LIU_MUTE_ON_LOS

DS3_TECT3_FRAMING_NONE
DS3_TECT3_FRAMING_M13
DS3_TECT3_FRAMING_CBITPARITY
```

pError Error code for the function (valid only if return value indicates the function failed).

Return Value

Success or failure of the function (0 = Fail, 1 = Success).

Notes

Contains an implicit call to DirectDSP API DSAssignBoard().

4. InitCtBus(UINT uMode, int nDspList, int nNumProcessors, KN_ERROR* const pError)

Description

Initialize CTBus on the SigC641x module.

Example Usage

```
if (!InitCtBus(DS_ICB_RESET | DS_ICB_LOGIC_LOOPBACK, nDspList, nNumProcessors,
               &knError))

printf("Problem with InitCtBus, knError = %d\n", knError);
```

Existing Code

```
appTest.cpp (look for “Enable TDM / CTbus registers on SigC641x module,…” comment)
```

Parameters

uMode Mode to initialize CTBus interface on the DSP module. One or more of the following mode values defined in tdmlib.h may be combined:

```
DS_ICB_RESET
DS_ICB_LOGIC_LOOPBACK
DS_ICB_CONNECT_DSP
```

The DS_ICB_RESET value should only be used if the CTBus interface should be reset. If used, then Reset is performed first, before other attributes, if specified, are applied.

The DS_ICB_LOGIC_LOOPBACK attribute will cause CTBus streams to be looped back inside FPGA logic on the SigC641x module. In this case a) loopback is essentially a “wire” connection with no delay, and b) no TDM time-slot data will reach the DSP farm.

nDspList A bitfield list of DSPs on the DSP module that should be internally connected to the CTBus. Each bit specifies a DSP, 1=enable, 0=disable, DSP0 starts at bit 0 (lsb). Serial ports McBSP0 and McBSP1 on each DSP are each connected to two (2) CTBus streams, or a physical pair of lines, the first line defined as ingress (data received by the DSP), and the second line defined as egress. McBSP0 connections start at stream 0, and McBSP1 connections start at stream 16. At a default CTBus clock rate of 8.192 MHz and frame rate of 8 kHz, each stream carries 128 time-slots. Therefore, each DSP has two (2) sets of full-duplex 1024 channel TDM connections to the CTBus. Normally, the carrier board is configured such that DS3 or OC3 module TDM data is mapped to CTBus streams 0-15 and H.110 TDM data is mapped to CTBus streams 16-31, using APIs InitDs3Module() and InitH110(), respectively.

The CTBus clock rate may be increased to 16.384 MHz to increase one or both sets of TDM data mappings to 2048 channels, or up to 8192 total time-slots accessible by the DSP farm.

nNumProcessors Number of processors on the SigC641x module. This value can range from 1 to 8; typical module configurations found in deployment are 4 DSPs and 8 DSPs.

pError Error code for the function (valid only if return value indicates the function failed).

Return Value

Success or failure of the function (0 = Fail, 1 = Success).

5. DSWriteDs3Device(HBOARD hBoard, DS3DEVICE uDevice, UINT uReg, UINT uData)

DSReadDs3Device(HBOARD hBoard, DS3DEVICE uDevice, UINT uReg)

Description

Provides direct access to devices on the wan256T3 DS3 module.

Example Usage

```
DSWriteDs3Device(hBoardDs3, DS3_TECT3, TEMUX_DS3_TRAN_CFG,
                 0x0d & tect3_reg_data_tran_cfg);

tect3_reg_data_tran_cfg = DSReadDs3Device(hBoardDs3, DS3_TECT3,
                                          TEMUX_DS3_TRAN_CFG);
```

Existing Code

```
appTest.cpp (look for “use masks needed to ensure unused bits...” comment)

appTest.cpp (look for “set TECT3 framer registers as needed” comment)
```

Parameters

hBoard Handle to DS3 module, assigned by previous call to InitDs3Module() or DSAssignBoard().

uDevice Device type, following constants and attributes are allowed (defined in tdmlib.h):

```
DS3_MUSYCC
DS3_TEMUX
DS3_TECT3
DS3_T8110
DS3_CPLD

DS3_REG8     /* attributes:  read/write one byte, read/write 4 bytes */
DS3_REG32
```

uReg Register address (offset) to read or write. Allowable constants are defined in temux.h.

uData Data to write to device register (only applicable to DSWriteDs3Device).

Return Value

Data read from device register (only applicable to DSReadDs3Device).

6. DSWriteH110Device(HBOARD hBoard, H110DEVICE uDevice, UINT uReg, UINT uData)

DSReadH110Device(HBOARD hBoard, H110DEVICE uDevice, UINT uReg)

Description

Provides direct access to T8110 and CPLD device registers on the HW400c/2 board.

Example Usage

```
d = DSReadH110Device(m_hBoardH110, H110_T8110, T8110_MASTER_ENABLE);
d &= 0xf000;

DSWriteH110Device(m_hBoardH110, H110_T8110, T8110_MASTER_ENABLE, d);
```

Existing Code

```
dsplib.cpp (look for “enable local bus clocks and streams,…” comment)
```

Parameters

hBoard Handle to the board, assigned by previous call to InitH110() or DSAssignBoard().

uDevice Device type to read from or write to. The following constants are allowed (defined in tdmlib.h):

```
H110_T8110
H110_CPLD
```

uReg Register address (offset) to read or write. Available register constants are shown below (defined in tdmlib.h):

```
T8110_MASTER_ENABLE
T8110_DATAMEMORY_MODE
T8110_CLOCKREG_ACCESS_SELECT
T8110_DEVICE_ID_REGISTER
T8110_MAIN_INPUT_SELECTOR
T8110_APLL1_INPUT_SELECTOR
T8110_RESOURCE_DIVIDER
T8110_LREF_INPUT_SELECT
T8110_MASTER_OUTPUT_ENABLE
T8110_LSC01_SELECT
T8110_LSC23_SELECT
T8110_HBUS_RATE_GROUP_BA
T8110_HBUS_RATE_GROUP_DC
T8110_HBUS_RATE_GROUP_FE
T8110_HBUS_RATE_GROUP_HG
T8110_LBUS_RATE_GROUP_BA
T8110_LBUS_RATE_GROUP_DC
T8110_LBUS_RATE_GROUP_FE
T8110_LBUS_RATE_GROUP_HG

T8110_CONNECTION_MEMORY_BASE     /* T8110 connection memory base address */
```

uData Data to write to device register (only applicable to DSWriteH110Device).

Return Value

Data read from device register (only applicable to DSReadH110Device).